Университет ИТМО

Факультет ПИиКТ

Системы искусственного интеллекта

Лабораторная работа №5

Выполнила: Наумова Н.А.

Группа P33022

Преподаватель: Бессмертный И.А.

Санкт-Петербург

2020 г.

Цель работы:

решить задачу многоклассовой классификации, используя в качестве тренировочного набора данных - набор данных MNIST, содержащий образы рукописных цифр.

Номер в списке группы ИСУ = 12

Задание:

1. Используйте метод главных компонент для набора данных MNIST (train dataset объема 60000). Определите, какое минимальное количество главных компонент необходимо использовать, чтобы доля объясненной дисперсии превышала 0.80+12%10 = 0.82. Построить график зависимости доли объясненной дисперсии от количества используемых ГК.
2. Введите количество верно классифицированных объектов класса 12%9 = 3 для тестовых данных.
3. Введите вероятность отнесения 5 любых изображений из тестового набора к назначенному классу
4. Определите Accuracy, Precision, Recall or F1 для обученной модели.
5. Сделайте вывод

```
# импорт необходимых библиотек, загрузка данных из MNIST
!pip install --upgrade pip
!pip install --upgrade scikit-learn==0.23.0
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import classification_report

from keras.datasets import mnist
(X_train, y_train), (X_pred, y_pred) = mnist.load_data()

# разделение выборки на тестовую (30%)  и тренировочную (70%)
from sklearn.model selection import train test split
X train, X test, y train, y test = train test split(X train, y train,
test size=0.3, random state=2020)

# преобразование данных и уменьшение размерности
dim = 784 # 28*28
X train = X train.reshape(len(X train), dim)
X test = X test.reshape(len(X test), dim)
```

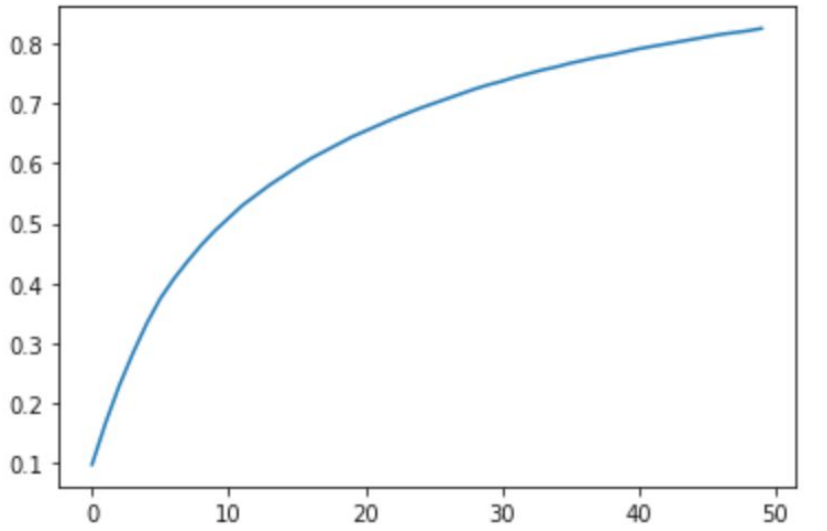
```
from sklearn.decomposition import PCA
pca = PCA(n components=50, svd solver='full')
modelPCA = pca.fit(X train)
X train = modelPCA.transform(X train)

#Найдем долю объясненной дисперсии в зависимости от кол-ва используемых ГК
explained variance = np.round(np.cumsum(pca.explained variance ratio ),3)
explained variance
```

```
array([0.098, 0.168, 0.23 , 0.284, 0.333, 0.375, 0.408, 0.437, 0.464,
       0.488, 0.509, 0.53 , 0.547, 0.564, 0.579, 0.594, 0.608, 0.62 ,
       0.632, 0.644, 0.654, 0.664, 0.674, 0.683, 0.692, 0.7  , 0.708,
       0.716, 0.724, 0.731, 0.737, 0.744, 0.75 , 0.756, 0.761, 0.767,
       0.772, 0.777, 0.781, 0.786, 0.791, 0.795, 0.799, 0.803, 0.807,
       0.811, 0.815, 0.818, 0.821, 0.825])
```

Видим, что необходимо использовать минимально 49 главных компонент, чтобы доля объясненной дисперсии превышала 0.82; выставим его выше.

```
# график
plt.plot(np.arange(50), explained variance, ls = '-')
```

```
[<matplotlib.lines.Line2D at 0x7f719c459198>]
```




```
# обучение многоклассового классификатора методом One-vs-All и дерево
# принятия решений
from sklearn.multiclass import OneVsRestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
tree = RandomForestClassifier(criterion='gini', min samples leaf=10,
max depth=20, n estimators=10, random state=2020)
clf = OneVsRestClassifier(tree).fit(X_train, y_train)
```

```
# применение полученного ранее преобразования метода главных компонент к
# тестовым данным
modelPCA = pca.fit(X test)
X test = modelPCA.transform(X test)
y pred = clf.predict(X test)
from sklearn.metrics import confusion matrix
CM = confusion matrix(y test, y pred)
CM
```

```
array([[1363,   0,  71,  78,   2,  49,  30,  49,  48,   3],
       [   0, 1860,  51,  11,  11,   6,  47,  49,  35,   5],
       [  54,  10, 986,  57,  46,  94, 357,  25, 111,  23],
       [  29,   5,  55, 1521,  16,  93,   7,  39,  80,  28],
       [   6,   9,  56,   7, 1348,  14,  43,  30,  70, 173],
       [ 104,   5, 181, 240,  51, 556,   4,  66, 356,  28],
       [  30,  19, 782,  30,  24,  22, 713,  25, 102,  19],
       [  59,  41,  17,   7,  42,  13,  84, 1493,  30, 100],
       [  35,  44, 108, 268,  64, 534,  35,  14, 569, 102],
       [  26,  16,  27,  47, 413,  37,  31, 175, 119, 933]])
```

```
CM[3][3]
```

```
1521
```

Число верно классифицированных объектов класса 3 - **1521.**

```
target names = ['class 0', 'class 1','class 2','class 3','class 4','class
5','class 6','class 7','class 8','class 9']
print(classification report(y test, y pred, target names=target names))
```

```
              precision    recall  f1-score   support

     class 0       0.80      0.81      0.80      1693
     class 1       0.93      0.90      0.91      2075
     class 2       0.42      0.56      0.48      1763
     class 3       0.67      0.81      0.73      1873
     class 4       0.67      0.77      0.71      1756
     class 5       0.39      0.35      0.37      1591
     class 6       0.53      0.40      0.46      1766
     class 7       0.76      0.79      0.78      1886
     class 8       0.37      0.32      0.35      1773
     class 9       0.66      0.51      0.58      1824

    accuracy                           0.63     18000
   macro avg       0.62      0.62      0.62     18000
weighted avg       0.63      0.63      0.62     18000
```

```
# вероятности
# 2-й элемент - класс 9
print(clf.predict_proba(X_train) [1] [y_pred[9]] )
```
0.7424156712046422

```
# 8192-й элемент - класс 3
print(clf.predict_proba(X_train) [8192] [y_pred[3]] )
```
0.7377543605972064

```
# 257-й элемент - класс 1
print(clf.predict_proba(X_train) [256] [y_pred[1]] )
```
0.1157121523252498

```
# 1025-й элемент - класс 4
print(clf.predict_proba(X_train) [1024] [y_pred[4]])
```
0.05633643200890198

```
# 4097-й элемент - класс 6
print(clf.predict_proba(X_train) [4096] [y_pred[6]] )
```
0.00507688388573749

Вывод: проанализировав значения accuracy, precision, recall заметим, что нейросеть дает не самые хорошие показатели точности - довольно точно она смогла распознавать только цифры класса “1” - единицы.
Выполнив данную лабораторную работу, я реализовала распознавание рукописных цифр на наборе данных MNIST.